Threat Detector®

System Administrator Manual

Version 1.1.0 | May 2022

Table of Contents

This is an interactive document. Click on any heading in the table of contents to activate the hyperlink

1. Introduction

This System Administrator Guide is a resource that provides information, tools, and techniques for deployment of the Threat Detector® service. The purpose of this guide is to be a reference for system administrators in order to use, run, and maintain the software.

1.1 About Threat Detector®

Threat Detector® is an interactive and seamless virus hash and URL scanning service. A series of malware engines are used to scan the entry and determine the malicious status of the hash or URL. The brains of the service are linked with the VirusTotal API to scan the user's entries.

Features that the service provides are the ability to enter hashes and URLs for scanning, viewing the malicious status of a hash/URL, viewing the status of the vendors for each hash or URL, and viewing hash and URL meta data. API calls are made to scan the entered hash and URLs in order to provide a reliable analysis for the user's benefit.

1.2 Audience

The software is intended for a range of users: do-it-yourself prosumers, small businesses, and/or large corporations. System Administrators are expected and assumed to have a basic understanding of using Linux distributions and working in a Linux terminal.

2. System Overview

This section dives into a complete overview of Threat Detector®.

2.1 Background

Threat Detector® uses a composite, yet trivial, integral setup as the makeup of the service. Components include VirusTotal API, Python, Flask, SQLite3, and Bootstrap.

Python-based Flask is used as a back-end to communicate with the Bootstrap-HTML framework on the front-end. An SQLite3 database is used to hold the data in the program. The back-end has all functionality to get the required properties (sha256, md5, sha1, etc…) from the database or API to display to the user. The VirusTotal API is called to get the results if the hash or URL has never been scanned before; otherwise, the results are pulled from the database. The back-end passes the results to the front-end to display to the user in an easy-to-read format.

The software contains three main python files, a database file, a templates folder, and a static folder. The templates folder holds the HTML pages rendered by the back-end, and the static folder holds images and CSS files for the HTML pages.

2.2 System Administrator Procedures

A system administrator for Threat Detector® needs to fulfill the following task:

- Installation of the service
- Launching and restarting the service
- Maintaining the service
- Using the service
- Database management
- Proper backup strategy (3-2-1 strategy recommended)
- Troubleshooting the service
- Fine-tuning system parameters for smooth execution

These tasks are required for the system administrator as they encompass proper technique to fully set up, run, maintain, troubleshoot, and safeguard the service and its systems.

2.3 Hardware and Software Requirements

The following system requirements are guidelines to have a stable experience with Threat Detector®.

The hardware processing power needed is very minimal. Essentially, the minimum requirements are very similar as to what is required to run a Linux distribution:

- `CPU: 1.0 GHz 64-bit or higher`
- `RAM: 1 GB or greater`
- `Disk Space: 5 GB or greater`

The software has been tested on Linux distributions only. A device with a Linux distribution installed suffices for execution. Python 3.6 or higher is recommended with pip3 installed as well. A `requirements.txt` is mentioned in the installation process which specifies the rest of the packages needed.

Other distributions are not supported by this documentation or future releases; however, there are ways to brute-force the software for other operating systems, so it is up to the system administrator to run it on non-supported systems.

3. Administrative Procedures

As a system administrator, there are important procedures that must be followed to ensure smooth operation of Threat Detector®.

3.1 Installation

A `requirements.txt` is provided and is run by the `threatdetector_app.sh` executable. A virtual environment is not needed, but recommended. Now, you can clone the repo into a directory.

After cloning the repo, navigate to the folder; you should see a file called `threatdetector_app.sh`. Enter the following command to run the application: `./threatdetector_app.sh`

This installs `pip3` if it is not already installed as well as the necessary dependencies to run the application. Then, navigate to the `src/` folder and run the following command:

```
python3 app.py
```

This starts Flask and the application, giving you an address to enter into your browser to view the website. The address should be: `http://127.0.0.1:5000/`. However, given the unpredictability of computers, it may be different for some odd reason, so check the output, specifically where it says `"Running on...."` to see the address.

NOTE: If the `threatdetector_app.sh` does not work, and you are unable to launch the application, try running `./threatdetector_app_backup.sh`. This executable creates an environment, install Flask and necessary dependencies, and run the application. This executable and application has been tested on Linux distributions only; operating distribution commands varies. Use Linux for the best experience!

3.2 Routine Tasks

A system administrator for Threat Detector® needs to perform just a few routine maintenance and task to ensure the service is working as intended.

The terminal in which Threat Detector® is running in needs to be monitored in case any error messages pop up in the output, or the program unexpectedly quits.

Nonetheless, the database must be periodically checked to see if any incorrect data has leaked into the tables to ensure accurate extraction from the database.

3.3 Periodic Administration

Ther are some vital periodic tasks that need to be performed to safeguard Threat Detector® and enable it to function to its fullest capacity.

Threat Detector® software needs to be rebooted at least once per week to ensure the program has not stalled or crashed during operation. A fresh reboot of Threat Detector® allows for a clean slate during operation.

General server and computer maintenance, such as component upgrades and server restarts, are needed to ensure no hardware malfunctions.

Finally, weekly backups of the database file should be scheduled in the event of a software or hardware failure.

3.4 User Support

Because of the small foot print of Threat Detector®, there is no on-demand support for this service. However, you can email one of the developers if you have any questions/concerns, bug fixes, or future ideas:

Mark Biegel | Software Developer
mbiegel1@umbc.edu

4. Troubleshooting

Troubleshooting Threat Detector® is relatively simple, as there are few moving parts as part of the service.

4.1 Dealing with Error Messages and Failures

Threat Detector® can produce a few error messages during operation if something has gone wrong with the service. All of these error messages are printed to in the terminal window containing the runtime for Threat Detector®.

Most of the error message pertain to the invalid requests that the user has entered; a system administrator does not have to deal with these, as Threat Detector® provides feedback to the user about the error.

Other error messages include fatal add and retrieve requests from the database and improper API calls to VirusTotal®. Improper API calls could be a result of a bad request on VirusTotal's end, or it could be a query limit of 500 searches has been reached. To deal with improper API calls, restart the service. If the problem persists, a query limit error could be a result; waiting 24 hours fixes this issue.

If there are any failures with Threat Detector®, close the service and restart it. If issues and errors persist, reboot the server/device Threat Detector® is running on. If errors persist, check for hardware inconsistencies and issues of the device Threat Detector® is running on.

4.2 Known Bugs and Limitations

In the current version, Threat Detector contains a few bugs that are to be improved upon in future releases:

1. The current query limit for searches is 500 queries a day, but the user may not know if they've reached a query limit since they are redirected to a page saying "Invalid Entry". Currently, the code does not have a way of determining if a query limit has been reached. This may upset the user because the user may think they are entering an incorrect and may get frustrated with the service since it constantly gives an "Invalid Entry" response. The users may, then, complain to the system administrator

about the issue, and the system administrator may try to solve the problem (i.e rebooting the service or the device) with no luck. Keep this in mind when an error that is not fixed with a reboot arises.

2. Furthermore, some formatting bugs with button placement of detection page exist when the browser window is not maximized; these bugs exist in `second_page_hash_malware_positive.html, second_page_hash_malware_negative.html, second_page_URL_malware_negative.html, and second_page_URL_malware_negative.html` as they pertain to the front-end HTML code. When browser is not maximized, the button layout for `Detection`, `File Meta Data`, and `Malicious Status` causes the `Malicious Status` button to be larger than the `Detection` and `File Meta Data` buttons. Because of the unintuitive look and feel of the user interface these buttons create, this could cause confusion with the user and overall approval of the service could decrease.

3. Another limitation with the current API Key being used from VirusTotal. Threat Detector® uses VirusTotal's Free API key and not Premium API key, meaning Threat Detector® can't scan every URL since some URLs require premium access to VirusTotal. Again, there is a 500 query-limit for new searches per day, and an invalid response page is displayed if this limit is reached. This limitation is can cause confusion with the user and leave them frustrated with the service. The users may, then, complain to the system administrator about the issue, and the system administrator may try to solve the problem with no luck.

4. Another bug is with scanning URLs using VirusTotal's API. Given the free API version, VirusTotal is very picky about what is receives from the user. Anything other than a domain name will result in VirusTotal sending no data to Threat Detector®. This results in a false invalid request reading which redirects the user to the invalid request page even though they haven't entered incorrect data. A Premium API Key most likely solves this problem. As of now, this can cause frustration among the users that try to scan valid URLs, resulting in an overall disapproval of the service. A note is displayed to the user on the invalid request page, giving them options to troubleshoot their error.